Sistemas lineares

Ricardo Biloti

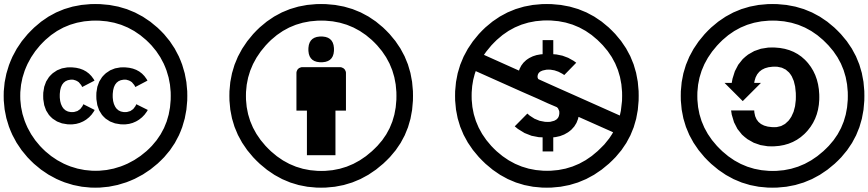
biloti@unicamp.br

Álgebra Linear Computacional

2S/2021

www.ime.unicamp.br/~biloti

Licença



Operações básicas em ALC

- Produto escalar, $x^T y$
- *saxpy*, $\alpha x + y$
- Produto matriz-vetor, Ax
- Produto matriz-matriz, AB

Os algoritmos que surgem em Álgebra Linear computacional podem ser estruturados em torno de algumas operações elementares. Essas operações são classificadas em operações de nível 1, 2 ou 3, dependendo da complexidade delas em termos da dimensão dos termos envolvidos.

As operações de produto interno e *saxpy* são operações de nível 1, uma vez que seu custo computacional é proporcional a n, a dimensão dos vetores.

Um produto matriz-vetor é uma operação de nível 2, dadoq ue seu custo computacional é proporcional a n^2, se a matriz for quadrada de ordem n.

De forma análoga, um produto matriz-matriz é uma operação de nível 3.

BLAS

Basic Linear Algebra Subprograms – BLAS [http://www.netlib.org/blas] – é um pacote de rotinas que implementa, de forma eficiente e portável, operações básicas com vetores e matrizes.

A BLAS é a base para o desenvolvimento de algoritmos e programas de alto nível em Álgebra Linear.

Implementações da BLAS podem ser fornecidas pelos desenvolvedores de processadores. Essas implementações são otimizadas para a arquitetura do processador, de modo a ser o mais eficiente possível.

O projeto ATLAS, por exemplo, testa diversos parâmetros que podem ser ajustados na BLAS, de modo a escolher heuristicamente qual a melhor configuração para um computador particular.

Algoritmos construídos tendo a BLAS como base, beneficiam-se desses ajustes.

Alguns softwares que dependem da BLAS: LAPACK, Matlab, Octave, R, NumPy, SciPy, etc. No Debian, 282 pacotes dependem explicitamente da BLAS, 312 dependem da NumPy, 288 dependem da LAPACK, por exemplo.

Produto escalar

$$x^T y : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

- $s \leftarrow 0$
- Para $i = 1 : n$
  - $s \leftarrow s + x_i y_i$

Custo: $\sum_{i=1}^{n} 2 = 2n = \mathcal{O}(n)$

- $s \leftarrow 0$
- $m \leftarrow \mathrm{mod}(n, 5)$
- Para $i = 1 : m$
  - $s \leftarrow s + x_i y_i$
- Para $i = m + 1 : 5 : n$
  - $s \leftarrow s + x_i y_i + x_{i+1} y_{i+1} + x_{i+2} y_{i+2} + x_{i+3} y_{i+3} + x_{i+4} y_{i+4}$

A versão implementada na BLAS acumula mais parcelas por cada iteração do loop para que o *overhead* gerado pelas instruções de controle do loop seja proporcionalmente menor.

saxpy

$$\alpha x + y : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$

- Para $i = 1 : n$
  - $y_i \leftarrow y_i + \alpha x_i$

Custo: $\sum_{i=1}^{n} 2 = 2n = \mathcal{O}(n)$

- $s \leftarrow 0$
- $m \leftarrow \mathrm{mod}(n, 4)$
- Para $i = 1 : m$
  - $y_i \leftarrow y_i + \alpha x_i$
- Para $i = m + 1 : 4 : n$
  - $y_i \leftarrow y_i + \alpha x_i$
  - $y_{i+1} \leftarrow y_{i+1} + \alpha x_{i+1}$
  - $y_{i+2} \leftarrow y_{i+2} + \alpha x_{i+2}$
  - $y_{i+3} \leftarrow y_{i+3} + \alpha x_{i+3}$

Produto matriz-vetor

$$\begin{bmatrix} 1 & 3 & 5 \\ 0 & 2 & 7 \\ 1 & 9 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 6 \end{bmatrix} = \begin{bmatrix} 40 \\ 46 \\ 60 \end{bmatrix}$$

Algoritmo orientado a linhas:

- Para $i = 1 : m$
  - $b_i \leftarrow a_i^T x$ (produto escalar) Custo computacional: $2nm$

⚠ a_i representa a i-ésima linha de A

Repareq que este algoritmo realiza $m(2n-1) = \mathcal{O}(mn)$ operações de ponto flutuante.

Produto matriz-vetor

$$\begin{bmatrix} 1 & 3 & 5 \\ 0 & 2 & 7 \\ 1 & 9 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 6 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 3 \\ 2 \\ 9 \end{bmatrix} + 6 \begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix} = \begin{bmatrix} 40 \\ 46 \\ 60 \end{bmatrix}$$

Algoritmo orientado a colunas:

- $b \leftarrow 0$
- Para $i = 1 : n$
  - $b \leftarrow x_i a_i + b$ (saxpy)

Custo computacional: $2nm$

⚠ a_i representa a i-ésima coluna de A

Assim como o algoritmo anterior. este algoritmo também realiza $m(2n-1) = \mathcal{O}(mn)$ operações de ponto flutuante, apenas trocando a ordem com a qual elas são realizadas.

Qual algoritmo escolher?

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 0 & 2 & 7 \\ 1 & 9 & 1 \end{bmatrix}$$

Em Fortan, A é armazenada na memória como (use a versão saxpy)

1 0 1 3 2 9 5 7 1

Em C, A é armazenada na memória como (use a versão produto interno)

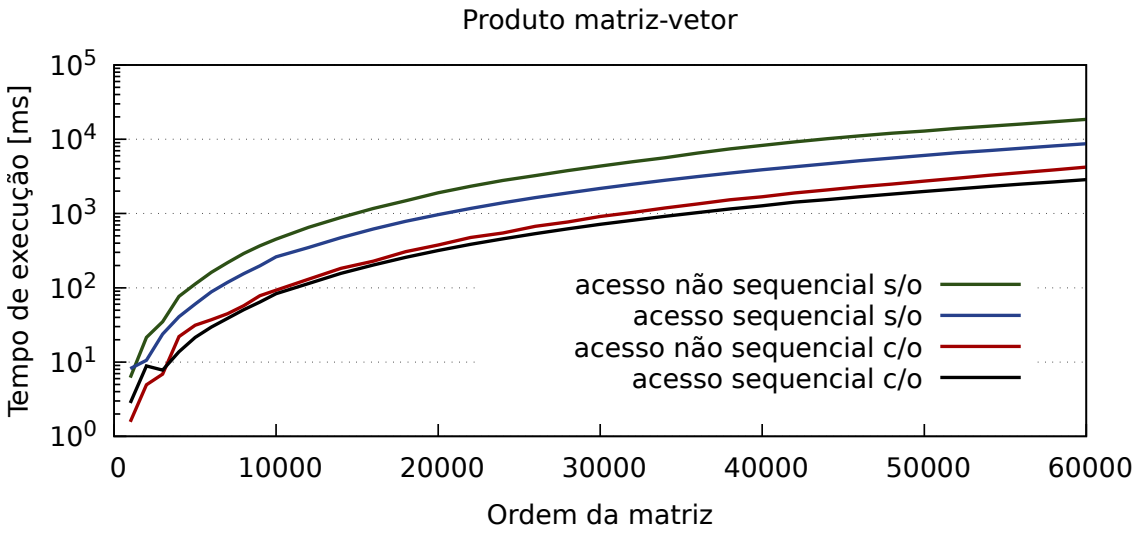
1 3 5 0 2 7 1 9 1

Se os dois algoritmos consomem o mesmo número de operações, qual dos dois escolher? Há diferença?

O tempo de execução de um algoritmo não relaciona-se apenas com número de operações de ponto flutuante executadas, mas também com a movimentação de dados entre a memória principal do computador e a memória do processador, onde as contas são de fato executadas.

Algoritmos que acessam a memória sequencialmente são mais economicos do ponto de vista de carregamento de dados para a memória do processador.

Não custa conferir...



Neste teste rodamos um programa que computa o produto matriz vetor por dois algoritmos diferentes, um deles que acessa os elementos da matriz de forma sequencial na memória e outro que os acesso de forma não sequencial. Os tempos de execucão, em milissegundos, foram registrados em função da ordem da matriz, que variou entre 1.000 e 60.000. Além disso, o mesmo experimento foi repetido usando-se flags de otimização do compilador.

Repare que a versão com acesso sequencial sempre foi mais rápida que a versão sem acesso sequencial. Além disso, permitir que o compilador otimize o código também ajuda bastante no tempo de execução. O que será que o compilador faz com um código tão simples, que consegue melhorá-lo?

Produto matriz-matriz

$$AX = A \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ax_1 & Ax_2 & \cdots & Ax_p \\ | & | & & | \end{bmatrix} = B$$

- Produto matriz-matriz é uma sequência de produtos matriz-vetor
- Ambos os algoritmos para produto matriz-vetor podem ser usados
- Custo computacional $2mnp$

Particionamento em blocos

$$\left[\begin{array}{cc|c} 1 & 0 & 2 \\ 3 & 1 & 1 \\ \hline 4 & 3 & 1 \end{array}\right] \left[\begin{array}{c|cc} 0 & 2 & 0 \\ 1 & 7 & 3 \\ \hline 0 & 1 & 1 \end{array}\right] = \left[\begin{array}{c|cc} 0 & 4 & 2 \\ 1 & 14 & 4 \\ \hline 3 & 30 & 10 \end{array}\right]$$

- $\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} 0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
- $\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 7 & 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 14 & 4 \end{bmatrix}$
- $\begin{bmatrix} 4 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1 \cdot 0 = 3$
- $\begin{bmatrix} 4 & 3 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 7 & 3 \end{bmatrix} + 1 \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 30 & 10 \end{bmatrix}$

Particionamento em blocos

$$AX = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c|c} X_{11} & X_{12} \\ \hline X_{21} & X_{22} \end{array}\right] = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right] = B$$

$$\begin{cases} A_{11}X_{11} + A_{12}X_{21} &= B_{11} \\ A_{11}X_{12} + A_{12}X_{22} &= B_{12} \\ A_{21}X_{11} + A_{22}X_{21} &= B_{21} \\ A_{21}X_{12} + A_{22}X_{22} &= B_{22} \end{cases}$$

⚠ O particionamento deve ser feito para que as dimensões façam sentido nas operações de produto

Norma

Sejam V um espaço vetorial, $x, y \in V$, arbitrários, e $\alpha \in \mathbb{R}$.
Se f é uma função $f : V \to \mathbb{R}$ satisfazendo

- $f(x) \geq 0$
- $f(x) = 0 \iff x = 0$
- $f(\alpha x) = |\alpha| f(x)$
- $f(x + y) \leq f(x) + f(y)$ (desigualdade triangular)

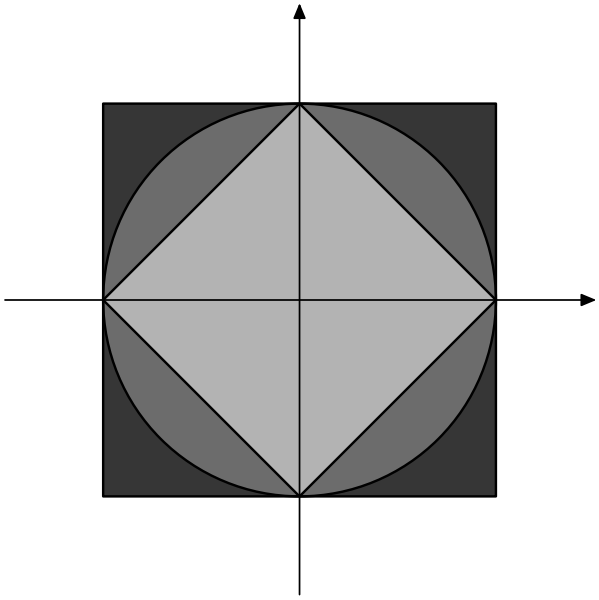
então f dita uma norma para V e denotada por $\|\cdot\|$.

Normas em $\mathbb{R}^n$

Se $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$, algumas normas são

- Norma Euclidiana: $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
- Norma 1: $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$
- Norma p: $\|x\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{(1/p)}$
- Norma infinito: $\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$

$\|x\| \leq 1$



Desigualdade de Cauchy-Schwartz

Teorema

Para quaisquer x, $y \in \mathbb{R}^n$,

$$|x^T y| \leq \|x\|_2 \, \|y\|_2$$

Para $u = x + ty$, $\|u\|_2 \geq 0$, para todo $t \in \mathbb{R}$, isto é,

$$\|u\|_2^2 = \sum_{i=1}^{n} (x_i + ty_i)^2 = \sum_{i=1}^{n} x_i^2 + 2t \sum_{i=1}^{n} x_i y_i + t^2 \sum_{i=1}^{n} y_i^2 = \|x\|_2^2 + 2t(x^T y) + t^2 \|y\|_2^2 = p(t)$$

Como o polinômio p é sempre positivo, ele não pode ter duas raízes distintas. Logo, seu discriminante é menor ou igual a zero, ou seja,

$$\left[2(x^T y)\right]^2 - 4\|x\|_2^2 \, \|y\|_2^2 \leq 0$$

Tomando a raiz quadrada acima, chegamos no resultado.

Normas matriciais

Sendo $\mathbb{R}^{m\times n}$ um espaço vetorial, também é possível definir normas para o conjunto das matrizes.

Se $\|\cdot\|$ é uma norma vetorial, podemos definir uma norma matricial, induzida pela norma vetorial, por

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

Propriedade: $$\frac{\|Av\|}{\|v\|} \leq \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \|A\| \;\Rightarrow\; \|Av\| \leq \|A\| \cdot \|v\|$$

Normas matriciais

- $\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}$ (norma de Frobenius)
- $\|A\|_1 = \max_j \|a_j\|_1$, onde a_j são as colunas de A
- $\|A\|_\infty = \max_i \|a_i\|_1$, onde a_i são as linhas de A
- $\|A\|_2 = \sqrt{\lambda}$, onde λ é o maior autovalor em de $A^T A$

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 0 & 4 & 5 \\ 1 & 5 & 2 \end{bmatrix}, \qquad \|A\|_F = \sqrt{82}, \quad \|A\|_1 = 12, \quad \|A\|_\infty = 9, \quad \|A\|_2 = 8.6834$$

Matrizes ortogonais

Uma matriz quadrada Q é dita ortogonal se

$$Q^T Q = I$$

- $Q^T = Q^{-1}$
- $QQ^T = I$
- As colunas de Q ortonormais entre si
- As colunas de Q formam um base ortonormal para $\mathbb{R}^n$

Propriedades

Se Q é uma matriz ortogonal, então

- $|\det Q| = 1$ (exercício)
- $\langle Qx, Qy\rangle = x^T Q^T Qy = x^T y = \langle x, y\rangle$
- $\|Qx\|_2 = \|x\|_2$

O problema

Queremos resolver

$$Ax = b, \qquad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^n$$

Este problema é simples se A for

- I, a matriz identidade
- D, uma matriz diagonal
- T, uma matriz triangular

O principal problema de Álgebra Linear é a resolução de sistemas lineares. Esse problema é simples quando a matriz de coeficientes tem alguma *estrutura* específica.

Sistema triangular

$$\begin{cases} a_{11}\,x_1 & & & & & & & = & b_1 \\ a_{21}\,x_1 & + & a_{22}\,x_2 & & & & & = & b_2 \\ a_{31}\,x_1 & + & a_{32}\,x_2 & + & a_{33}\,x_3 & & & = & b_3 \\ a_{41}\,x_1 & + & a_{42}\,x_2 & + & a_{43}\,x_3 & + & a_{44}\,x_4 & = & b_4 \end{cases}$$

$$\begin{aligned} x_1 &= (b_1)/a_{11} \\ x_2 &= (b_2 - a_{21}\,x_1)/a_{22} \\ x_3 &= (b_3 - a_{31}\,x_1 - a_{32}\,x_2)/a_{33} \\ x_4 &= (b_4 - a_{41}\,x_1 - a_{42}\,x_2 - a_{43}\,x_3)/a_{44} \end{aligned}$$

$$x_k = \frac{b_k - \sum_{j=1}^{k-1} a_{kj}\,x_j}{a_{kk}}, \quad k = 1, 2, \ldots, n$$

Sistema triangular por blocos

$$Gx = b$$

$$\left[\begin{array}{c|ccc} \times & 0 & 0 & 0 \\ \hline \times & \times & 0 & 0 \\ \times & \times & \times & 0 \\ \times & \times & \times & \times \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \\ x_3 \\ x_4 \end{array}\right] = \left[\begin{array}{c|c} g_{11} & 0 \\ \hline \hat{g} & \hat{G} \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline \hat{x} \end{array}\right] = \left[\begin{array}{c} b_1 \\ \hline \hat{b} \end{array}\right]$$

$$\left\{\begin{array}{ll} g_{11}x_1 & = b_1 \\ x_1\hat{g} + \hat{G}\hat{x} & = \hat{b} \end{array}\right. \quad \Longrightarrow \quad \left\{\begin{array}{ll} x_1 & = b_1/g_{11} \\ \hat{G}\hat{x} & = \hat{b} - x_1\hat{g} \end{array}\right.$$

Algoritmo orientado por colunas

$$\begin{cases} x_1 & = b_1/g_{11} \\ \hat{G}\hat{x} & = \hat{b} - x_1\hat{g} \end{cases}$$

- Para $j = 1 : n$
  - Se $g_{jj} = 0$, a matriz é singular. FIM.
  - $x_j \leftarrow b_j/g_{jj}$
  - Para $i = j + 1 : n$ (saxpy: $\hat{b} \leftarrow -x_j\hat{g} + \hat{b}$)
    - $b_i \leftarrow b_i - x_j g_{ij}$

Custo: $\sum_{j=1}^{n} 1 + 2(n-j) = (1+2n)n - 2\sum_{j=1}^{n} j = (1+2n)n - 2\frac{n(n-1)}{2} = n^2$

Decomposição de Cholesky

Dado A, $n \times n$, se A for

- simétrica ($A^T = A$)
- definida positiva ($x^T Ax > 0$, para todo $x \neq 0$)

então existe G triangular inferior, com diagonal positiva, tal que

$$A = GG^T$$

Exemplo

$$\begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 2 \\ 2 & 2 & 18 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & -1 & 4 \end{bmatrix} \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 4 \end{bmatrix}$$

Onde surgem

Aplicações em que surgem matrizes simétricas e definidas positivas

- Discretizações da equação de Laplace
- Problemas em engenharia
- Problemas de otimização
- Problemas de quadrados mínimos

Decomposição de Cholesky em blocos

$$A = \left(\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right) = \left(\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right) \left(\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right) = GG^T$$

Teorema

Se A é s.d.p. e G é o fator de Cholesky A, então $\hat{A}$ é s.d.p. e $\hat{G}$ é o fator de Cholesky de $\hat{A}$.

Que $\hat{A}$ é simétrica e $\hat{G}$ é triangular inferior é verificável simplesmente pela definição dessas matrizes. Se $\hat{x} \neq 0$, então

$$\hat{x}^T \hat{A} \hat{x} = [\hat{x}^T \; 0] \, A \begin{bmatrix} \hat{x} \\ 0 \end{bmatrix} > 0,$$

visto que A é s.d.p. Logo $\hat{A}$ é s.d.p. Como o fator de Cholesky é único, este só pode ser $\hat{G}$.

Decomposição de Cholesky em blocos

$$A = \left(\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right) = \left(\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right) \left(\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right) = GG^T$$

Logo

$\hat{A} = \hat{G}\hat{G}^T$	Conhecido $\hat{G}$, fator de Cholesky de $\hat{A}$,
$a = \hat{G}g$	calcule g, resolvendo $\hat{G}g = a$
$\alpha = g^T g + \gamma^2$	defina $\gamma = \sqrt{\alpha - g^T g}$

⚠ Como $\gamma = \sqrt{\alpha - g^T g}$, será que $\alpha - g^T g > 0$?

Decomposição de Cholesky em blocos

$$A = \left(\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right) = \left(\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right) \left(\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right) = GG^T$$

Como A é s.d.p., para qualquer $x \neq 0$, $x^T Ax > 0$, tomando $x^T = [\hat{x}^T \ \mu]$, temos

$$\begin{aligned} 0 < x^T Ax &= \hat{x}^T \hat{A} \hat{x} + 2\mu \hat{x}^T a + \alpha\mu^2 \\ &= \hat{x}^T \hat{G} \hat{G}^T \hat{x} + 2\mu \hat{x}^T \hat{G} g + \alpha\mu^2 \\ &= g^T g - 2g^T g + \alpha = -g^T g + \alpha \end{aligned}$$

Para $\hat{G}^T \hat{x} = -g$ e $\mu = 1$.

Exemplo

$$\left[\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right] = \left[\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right] \left[\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right]$$

$$\begin{aligned} \hat{A} &= \hat{G}\hat{G}^T \\ a &= \hat{G}g \\ \alpha &= g^T g + \gamma^2 \end{aligned}$$

$$\begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 2 \\ 2 & 2 & 18 \end{bmatrix} = \begin{bmatrix} \times & & \\ \times & \times & \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \end{bmatrix}$$

Exemplo

$$\left[\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right] = \left[\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right] \left[\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right]$$

$$\begin{aligned} \hat{A} &= \hat{G}\hat{G}^T \\ a &= \hat{G}g \\ \alpha &= g^T g + \gamma^2 \end{aligned}$$

$$\begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 2 \\ 2 & 2 & 18 \end{bmatrix} = \begin{bmatrix} 2 & & \\ 3 & 1 & \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} 2 & 3 & \times \\ & 1 & \times \\ & & \times \end{bmatrix}$$

$$\begin{aligned} 4 &= 2 \cdot 2 \\ 6 &= 2 \cdot g \Rightarrow g = 3 \\ 10 &= 3 \cdot 3 + \gamma^2 \Rightarrow \gamma = 1 \end{aligned}$$

Exemplo

$$\left[\begin{array}{c|c} \hat{A} & a \\ \hline a^T & \alpha \end{array}\right] = \left[\begin{array}{c|c} \hat{G} & 0 \\ \hline g^T & \gamma \end{array}\right] \left[\begin{array}{c|c} \hat{G}^T & g \\ \hline 0^T & \gamma \end{array}\right]$$

$$\hat{A} = \hat{G}\hat{G}^T$$
$$a = \hat{G}g$$
$$\alpha = g^T g + \gamma^2$$

$$\begin{bmatrix} 4 & 6 & 2 \\ 6 & 10 & 2 \\ 2 & 2 & 18 \end{bmatrix} = \begin{bmatrix} 2 & & \\ 3 & 1 & \\ 1 & -1 & 4 \end{bmatrix} \begin{bmatrix} 2 & 3 & 1 \\ & 1 & -1 \\ & & 4 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 6 \\ 6 & 10 \end{bmatrix} = \begin{bmatrix} 2 & \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 & \\ 3 & 1 \end{bmatrix} g \Rightarrow g = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$18 = g^T g + \gamma^2 \Rightarrow \gamma = 4$$

Algoritmo

Seja A uma matriz de ordem n simétrica.

- Se $A(1,1) \leq 0$, não há decomposição de Cholesky. FIM
- $G(1,1) \leftarrow \sqrt{A(1,1)}$ — 1
- Para $i = 2 : n$
  - Resolva $G(1 : i-1, 1 : i-1)g = A(1 : i-1, i)$ — $(i-1)^2$
  - $G(i, 1 : i-1) \leftarrow g^T$
  - $s \leftarrow A(i,i) - g^T g$ — $2(i-1)$
  - Se $s \leq 0$, não há decomposição de Cholesky. FIM
  - $G(i,i) = \sqrt{s}$ — 1

Custo: $1 + \sum_{i=2}^{n}(i-1)^2 + 2(i-1) + 1 = \sum_{i=1}^{n-1} i^2 + \mathcal{O}(i) = \frac{n^3}{3} + \mathcal{O}(n^2)$

Exercício

Encontre a decomposição de Cholesky de

$$\begin{bmatrix} 16 & -4 & 8 \\ -4 & 2 & -1 \\ 8 & -1 & 14 \end{bmatrix}$$

Decomposição de Cholesky

Com o visto até agora, construímos um algoritmo e demonstramos construtivamente o seguinte resultado

Teorema

Seja A uma matriz de ordem n, simétrica e definida positiva. Então existe uma única matriz G, de ordem n, triangular inferior com diagonal positiva tal que

$$A = GG^T$$

Resolução de sistema linear

Se A é s.d.p. e G é seu fator de Cholesky, então

$$Ax = b \iff GG^T x = b \iff \begin{cases} Gy &= b \\ G^T x &= y \end{cases}$$

Tudo ou nada

Já vimos que o custo computacional da resolução de um sistema linear é $\mathcal{O}(n^3)$.

E se não houver recursos suficientes? Por exemplo, se não tiver...

- tempo,
- espaço,
- dinheiro?

Tudo ou nada

UNICAMP

Métodos diretos

Caso seja possível arcar com o custo total de um método direto, como eliminação do Gauss, decomposição LU ou Cholesky, ao final de um número finito de passo, a solução é conhecida, a menos de erros numéricos.

Métodos iterativos

Métodos iterativos obtém aproximações sucessivas da solução, cuja qualidade é *proporcional* à disponibilidade de recursos.

Modelo de método de iterativo

- Seja x_0 uma aproximação inicial
- $x_{k+1} = \phi(x_k)$, para $k = 0, 1, 2, \ldots$

Função de iteração e condição de ponto fixo

Se x^* é uma solução do problema, ϕ deve ter a propriedade de que

$$x^* = \phi(x^*)$$

Um método iterativo é dito de passo simples se para computar o iterando x_{k+1} é utilizada informação apenas da iteração k. De forma geral, todo método iterativo de passo simples pode ser representado por meio de uma função ϕ, dita função de iteração, que especifica como, a partir das informações do passo k, obter o novo iterando x_{k+1}.

Um exemplo de método iterativo de passo simples é o método de Newton. A função de iteração do método de Newton é

$$\phi(x) = x - \frac{f(x)}{f'(x)}.$$

Observe que se x^* for a solução procurada, então para que o método iterativo tenha chance de encontrá-la é necessário que $\phi(x^*) = x^*$. Isso significa que, se em algum momento um dos iterandos do método for de fato a solução do problema, todos os iterandos seguintes permanecerão sendo x^*.

Essa condição por si só não é suficiente, mas sim necessária. Tal condição recebe o nome de condição de ponto fixo.

Função de iteração

Splitting

Escrever A como $M + N$ é dito um splitting.

Se x^* é solução de $Ax = b$ e $A = M + N$, então

$$\begin{aligned} Ax^* = (M + N)x^* &= b \\ Mx^* &= -Nx^* + b \\ x^* &= -M^{-1}Nx^* + M^{-1}b \end{aligned}$$

Proposta:

$$\phi(x) \equiv Bx + c, \qquad B = -M^{-1}N,\ c = M^{-1}b$$

Para construir um método iterativo para aproximar a resolução do sistema linear $Ax = b$, é necessário então criar uma função de iteração ϕ. Como já vimos que a $\phi(x^*) = x^*$, uma proposta de função ϕ que cumpre essa condição é

$$\phi(x) = Bx + c,$$

onde B é construída a partir de algum *splitting* de A. Variando a forma com A é repartida na soma $M + N$, teremos diferentes funções de iteração.

B é dita a matriz de iteração do método.

Método de iterativo estacionário

$$\phi(x) = Bx + c, \qquad x^* = \phi(x^*)$$

- Seja x_0 uma aproximação inicial
- $x_{k+1} = \phi(x_k) = Bx_k + c$, para $k = 0, 1, 2, \ldots$

Este método é convergente?

Apenas o fato de ter construído uma função ϕ que satisfaz a condição de ponto fixo, não garante que um método iterativo a partir dessa função será bem sucedido.

Devemos então nos perguntar sob quais condições os iterandos gerados por esse método convergem para a solução do problema, ou seja, para x^*, solução do sistema linear.

Convergência

O método é convergente se $x_k \to x^*$, ou seja, se

$$\|e_k\| \equiv \|x_k - x^*\| \to 0$$

Dizer que x_k converge para x^* é dizer que o vetor diferença converge para o vetor nulo, o que pode ser medido pela norma do vetor.

Convergência

O método é convergente se $x_k \to x^*$, ou seja, se

$$\|e_k\| \equiv \|x_k - x^*\| \to 0$$

Como o erro varia em um passo do método?

$$\begin{aligned} e_k &= x_k - x^* \\ &= \phi(x_{k-1}) - \phi(x^*) \\ &= (Bx_{k-1} + c) - (Bx^* + c) \\ &= B(x_{k-1} - x^*) = Be_{k-1} \end{aligned}$$

Já definimos que para que um método seja convergente, o erro deve ir zero. Veja então como escrever o erro no passo (k) em função do erro no passo $(k-1)$.

Observe que $e_k = Be_{k-1}$.

Convergência

$$
\begin{aligned}
e_k &= Be_{k-1} \\
&= B\left(Be_{k-2}\right) \\
&= B^2\left(Be_{k-3}\right) \\
&= \cdots \\
&= B^k e_0
\end{aligned}
$$

$$
e_k = B^k e_0 \quad \Rightarrow \quad \|e_k\| \leq \|B\|^k \|e_0\|
$$

Se ao avançar um passo o vetor erro é multiplicado por B, é fácil perceber que, partindo do erro no iterando original, e_0, o erro no passo (k) será $e_k = B^k e_0$.

É necessário agora saber duas propriedades de normas de matrizes:

1. $\|AB\| \leq \|A\|\|B\|$
2. $\|Ax\| \leq \|A\|\|x\|$

Com essas propriedades, mostre que $\|B^k e_0\| \leq \|B\|^k \|e_0\|$.

Logo, para que o erro vá a zero, basta pedir que $\|B\| < 1$.

Modelo de método iterativo

- Seja x_0 uma aproximação inicial qualquer
- $x_{k+1} = \phi(x_k)$, para $k = 0, 1, 2, \ldots$
  com $\phi(x) = Bx + c$, tal que $x^* = \phi(x^*)$

$$\|B\| < 1 \quad \Rightarrow \quad x_k \to x^*$$

Assim chegamos à forma geral de método iterativo de passo simples para a resolução de sistema linear.

Uma pergunta recorrente é se $\|B\|$ precisa ser menor que 1 *em todas as normas*. A resposta é não. Se B tiver norma menor que 1, para alguma norma específica, isto será suficiente para garantir a convergência do método iterativo, naquela norma. Porém, se um método iterativo para sistema linear convergir em um determinada norma, o método será convergente em qualquer outra norma. Logo, basta demonstrar a convergência com alguma norma.

Por outro lado, caso B tenha alguma norma maior que 1 ainda há esperança de que o método produza uma sequência convergente, pois é possível que em outra norma a condição de convergência se cumpra.

Splittings

$$A = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}$$

$$A = \begin{pmatrix} & & & \\ \times & & & \\ \times & \times & & \\ \times & \times & \times & \end{pmatrix} + \begin{pmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix} + \begin{pmatrix} & \times & \times & \times \\ & & \times & \times \\ & & & \times \\ & & & \end{pmatrix}$$

$$A = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

Outra maneira de separar a matriz A em somas e considerando suas porções diagonal e triangulares inferior e superior.

Método de Jacobi

Escrevendo $A = D + (L + U) = D + (A - D)$,

$$\begin{aligned} Ax^* &= b \\ (D + A - D)x^* &= b \\ Dx^* &= (D - A)x^* + b \\ x^* &= D^{-1}(D - A)x^* + D^{-1}b \equiv \phi(x^*) \end{aligned}$$

Função de Iteração

$$\phi(x) = B_J x + c, \qquad B_J = D^{-1}(D - A), \quad c = D^{-1}b$$

O método de Jacobi é construído resolvendo-se um sistema diagonal.

Exemplo

$$\begin{pmatrix} 1.7 & -0.6 \\ -0.7 & 1.5 \end{pmatrix} x = \begin{pmatrix} 2.0 \\ -0.1 \end{pmatrix}$$

$$B_J = -\begin{pmatrix} 1.7 & \\ & 1.5 \end{pmatrix}^{-1} \begin{pmatrix} & -0.6 \\ -0.7 & \end{pmatrix} = \begin{pmatrix} 0.00 & 0.35 \\ 0.47 & 0.00 \end{pmatrix}$$

$$\|B_J\|_1 = \|B_J\|_\infty = 0.47$$

Observe que a matriz de iteração do método de Jacobi tem norma menor que 1, enquanto que já havíamos visto que para este mesmo sistema linear, a matriz de iteração do método de Richardson tinham norma maior que 1.

Convergência

$$B_J = -\begin{pmatrix} a_{11}^{-1} & & & & \\ & a_{22}^{-1} & & & \\ & & a_{33}^{-1} & & \\ & & & \ddots & \\ & & & & a_{nn}^{-1} \end{pmatrix} \begin{pmatrix} 0 & a_{12} & a_{13} & & a_{1n} \\ a_{21} & 0 & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & 0 & & a_{3n} \\ & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{pmatrix}$$

$$B_J = -\begin{pmatrix} 0 & a_{12}/a_{11} & a_{13}/a_{11} & & a_{1n}/a_{11} \\ a_{21}/a_{22} & 0 & a_{23}/a_{22} & \cdots & a_{2n}/a_{22} \\ a_{31}/a_{33} & a_{32}/a_{33} & 0 & & a_{3n}/a_{33} \\ & \vdots & & \ddots & \vdots \\ a_{n1}/a_{nn} & a_{n2}/a_{nn} & a_{n3}/a_{nn} & \cdots & 0 \end{pmatrix}$$

De forma geral, observando a matriz de iteração do método de Jacobi é possível vislumbrar um critério de convergência baseado diretamente nas entradas da matriz A original.

Critério das linhas

Para que $\|B_J\|_\infty < 1$, basta que

$$\|b^i\|_1 = \frac{|a_{i1}| + \cdots + |a_{i(i-1)}| + |a_{i(i+1)}| + \cdots + |a_{in}|}{|a_{ii}|} < 1,$$

para $i = 1, 2, \ldots, n$, ou seja

$$|a_{ii}| > \sum_{\substack{k=1 \\ k \neq i}}^{n} |a_{ik}|$$

O critério das linhas garante que o método de Jacobi será convergente se os elementos da diagonal da matriz dominarem os restantes dos elementos das linhas, ou seja, se em cada linha o elemento da diagonal em módulo foi maior que a soma dos elementos fora da diagonal, todos em módulo.

Matrizes com essa propriedade são ditas matrizes diagonalmente dominantes por linhas.

Exemplo

$$\begin{pmatrix} 4 & 2 & -1 \\ 0 & -3 & 1 \\ -2 & 2 & -6 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 2 & -1 \\ 6 & -3 & 1 \\ -1 & 2 & 7 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 2 & -1 \\ 2 & -3 & 7 \\ -1 & 2 & 0 \end{pmatrix}$$

A primeira matriz satisfaz o critério das linhas, visto que $4 > 2 + 1$, $3 > 0 + 1$, e $6 > 2 + 2$. Logo, seguramente o método de Jacobi aplicado a um sistema linear com esta matriz de coeficientes produzirá uma sequência convergente.

A segunda matriz não satisfaz o critério das linhas, visto que na segunda linha $3 < 6 + 1$.

A terceira matriz também não satisfaz o critério das linhas, dado que tanto a segunda como a terceira linhas violam a condição necessária ($3 < 2+7$ e $0 < 2+1$). Entretanto, permutando a segunda e a terceira linhas, a nova matriz sim satisfaz o critério das linhas. Portanto, é possível aplicar o método de Jacobi, desde que esta permutação seja feita antes.

Método de Gauss–Seidel

Escrevendo $A = (L + D) + U$,

$$\begin{aligned} Ax^* &= b \\ (L + D + U)x^* &= b \\ (L + D)x^* &= -Ux^* + b \\ x^* &= -(L + D)^{-1}Ux^* + (L + D)^{-1}b \equiv \phi(x^*) \end{aligned}$$

Função de Iteração

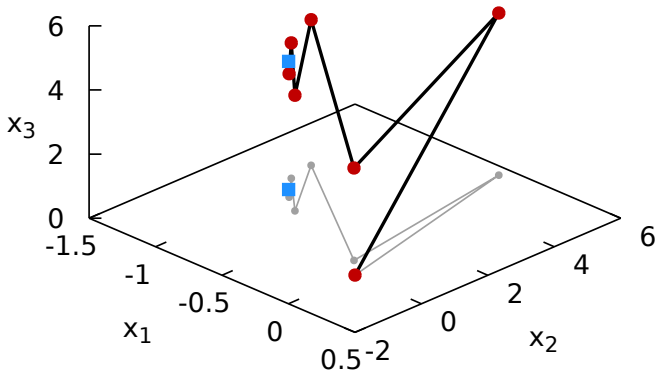
$$\phi(x) = B_{GS}x + c, \qquad B_{GS} = -(L + D)^{-1}U, \quad c = (L + D)^{-1}b$$

O método de Gauss–Seidel é construído resolvendo-se um sistema triangular inferior.

Também é possível obter um critério de convergência específico para o método de Gauss–Seidel, assim como foi feito para o método de Jacobi. Entretanto, a álgebra envolvida é mais tediosa e, portanto não o faremos aqui. A saber, se o método de Jacobi for convergente, o método de Gauss–Seidel também o será. Logo o critério das linhas também pode ser adotado aqui.

Gauss–Seidel em um exemplo

$$\begin{bmatrix} 3 & -1 & 1 \\ 1 & 2 & 2 \\ 3 & -2 & 5 \end{bmatrix} x = \begin{bmatrix} -1 \\ 11 \\ 13 \end{bmatrix}$$



Iteração de Gauss–Seidel × SOR

Para $i = 1, 2, \ldots, n$:

$$\sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} + a_{ii}\, x_i^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} = b_i$$

Para $i = 1, 2, \ldots, n$:

$$\sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} + a_{ii}\, \hat{x} + \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} = b_i$$

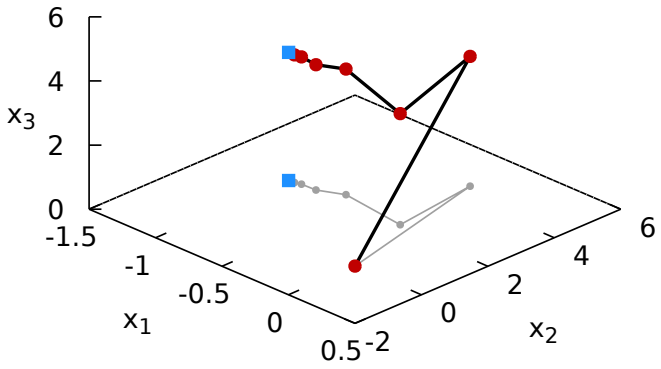
$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\hat{x} = x_i^{(k)} + \omega\left(\hat{x} - x_i^{(k)}\right)$$

UNICAMP

Método das Sobre Relaxações Sucessivas (SOR)

SOR em um exemplo

$$\begin{bmatrix} 3 & -1 & 1 \\ 1 & 2 & 2 \\ 3 & -2 & 5 \end{bmatrix} x = \begin{bmatrix} -1 \\ 11 \\ 13 \end{bmatrix}$$



Iteração de SOR

Para $i = 1, 2, \ldots, n$:

$$\sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} + a_{ii}\, \hat{x} + \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k)} = b_i$$

$$x_i^{(k+1)} = x_i^{(k)} + \omega \left(\hat{x} - x_i^{(k)} \right)$$

$$L x_{k+1} + D \hat{x} + U x_k = b$$

$$D \hat{x} = b - L x_{k+1} - U x_k$$

SOR

$$x_{k+1} = x_k + \omega\left(\hat{x} - x_k\right), \quad D\hat{x} = b - Lx_{k+1} - Ux_k$$

$$\begin{aligned} Dx_{k+1} &= Dx_k + \omega\left(b - Lx_{k+1} - Ux_k - Dx_k\right) \\ (D + \omega L)x_{k+1} &= (1-\omega)Dx_k - \omega Ux_k + \omega b \end{aligned}$$

$$\begin{aligned} x_{k+1} &= (D + \omega L)^{-1}\left[(1-\omega)D - \omega U\right]x_k + \omega(D + \omega L)^{-1}b \\ &= B_{SOR}x_k + c_{SOR} \end{aligned}$$

Convergência do SOR

UNICAMP

Teorema

Para que a iteração de SOR seja convergente é necessário que $\omega \in (0, 2)$.

Como $B_{SOR} = (D + \omega L)^{-1} [(1 - \omega)D - \omega U]$, se λ_i é autovalor de B_{SOR} então

$$\prod_{i=1}^{n} \lambda_i = \det\left((D + \omega L)^{-1}\right) \det\left((1 - \omega)D - \omega U\right) = \det\left(D^{-1}\right) (1 - \omega)^n \det(D) = (1 - \omega)^n$$

$$(1 - \omega)^n = \prod_{i=1}^{n} \lambda_i \leq \max |\lambda_i|^n = \rho\left(B_{SOR}\right)^n \quad \Rightarrow \quad \rho\left(B_{SOR}\right) \geq |\omega - 1|$$

Para que haja convergência, $\rho\left(B_{SOR}\right) < 1$. Portanto é necessário que $|\omega - 1| < 1$.

Demonstramos o teorema que garante convergência de métodos iterativos estacionários para sistemas lineares desde que alguma norma da matriz de iteração seja menor que 1.

Há um resultado mais forte que diz que há convergência se e somente se o raio espectral (maior autovalor em módulo) da matriz de iteração for menor que 1. Mostrar que ter raio espectral menor que 1 é uma condição necessária à convergência é simples. Por outro lado, a demonstração de que esta condição é suficiente é mais elaborada e foge ao escopo deste curso. A demonstração desse resultado pode ser vista na página 290 de J.W.Demmel, *Applied Numerical Linear Algebra*, SIAM, 1996.

Equação de Poisson em 2D

$$\Delta u = f(x,y) \quad \rightarrow \quad Ax = b, \quad A \in \mathbb{R}^{225\times 225}$$

Critério de parada: $\|x_{k+1} - x_k\|_\infty \leq 10^{-3}$

ω	k	$\|Ax-b\|_\infty$	
0.4	227	$8.0 \cdot 10^{-3}$	
0.6	197	$4.6 \cdot 10^{-3}$	
0.8	161	$3.0 \cdot 10^{-3}$	
1.0	129	$2.0 \cdot 10^{-3}$	$\leftarrow$ Gauss–Seidel
1.2	100	$1.3 \cdot 10^{-3}$	
1.4	74	$8.6 \cdot 10^{-4}$	
1.6	50	$5.2 \cdot 10^{-4}$	
1.8	26	$1.1 \cdot 10^{-3}$	

O operador Laplaciano é dado por

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

Quando a função f é identicamente nula, a equação de Poisson é dita *Equação de Laplace*.

Este operador é definido negativo. Para aplicar SOR, discretizamos por diferenças finitas a equação $-\Delta u = -f(x,y)$. Com isto, obteremos um sistema linear cuja matriz é definida positiva.

Note que as incógnitas são os valores de $u(x,y)$ nos pontos internos da malha de diferenças finitas. Portanto, se a malha tiver 15×15 pontos internos, teremos 225 incógnitas (nos pontos de borda da malha, usamos alguma condição de fronteira fornecida).

SOR e matrizes s.p.d.

UNICAMP

Teorema

Se A é uma matriz s.d.p. então o método SOR é convergente se e somente se $0 < \omega < 2$.

Corolário

Se A é uma matriz s.d.p. então o método Gauss–Seidel é convergente.

Outra abordagem

Construir método iterativos partindo de um splitting não é o único caminho.

Métodos estacionários $\times$ não estacionários

UNICAMP

- Métodos estacionários: há uma matriz de iteração fixa
  Jacobi, Gauss-Seidel, SOR
- Métodos não estacionários: não há uma matriz de iteração
  Gradiente ou Máxima descida, Gradientes conjugados, GMRES

Quadrática

Se A é uma matriz simétrica e definida positiva e $q : \mathbb{R}^n \to \mathbb{R}$ é dada por

$$q(x) = \frac{1}{2}x^T Ax - x^T b,$$

então

$$\nabla q(x) = Ax - b.$$

Logo, x_*, minimizador de q, é solução do sistema linear $Ax = b$.

Notação: $r \equiv b - Ax$ é dito o resíduo do sistema linear.

Exercício: verifique que, de fato, $\nabla q(x) = Ax - b$.

Direção do gradiente

Do Cálculo sabemos que o $\nabla q(x)$ aponta para direção de maior crescimento de q. Logo, q reduz na direção oposta ao gradiente.

Se x_k é uma aproximação para o mínimo de q, vamos definir a aproximação seguinte como

$$x_{k+1} = x_k + sr_k,$$

onde o escalar s é escolhido para que $q(x_{k+1})$ seja o menor possível.

Minimizando q na direção r_k

$$f(s) \equiv q(x_k + sr_k) = \frac{1}{2}(x_k + sr_k)^T A(x_k + sr_k) - (x_k + sr_k)^T b$$
$$= \frac{1}{2}\left(x_k^T Ax_k + 2sr_k^T Ax_k + s^2 r_k^T Ar_k\right) - x_k^T b - sr_k^T b$$

Logo,

$$f'(s) = r_k^T Ax_k + s\left(r_k^T Ar_k\right) - r_k^T b = r_k^T(Ax_k - b) + s\left(r_k^T Ar_k\right) = -r_k^T r_k + s\left(r_k^T Ar_k\right)$$

$$f'(s) = 0 \iff s = \frac{r_k^T r_k}{r_k^T Ar_k}$$

Método do gradiente

Sejam A uma matriz s.d.p. e $b \in \mathbb{R}^n$. Sejam x_0 uma aproximação inicial para a solução de $Ax = b$, $N \in \mathbb{N}$ e $\epsilon > 0$, dados.

- $k \leftarrow 0$
- $r_0 \leftarrow b - Ax_0$
- Enquanto $r_k^T r_k > \epsilon$ e $k \leq N$, faça
  - $s = (r_k^T r_k) / (r_k^T A r_k)$
  - $x_{k+1} \leftarrow x_k + s \cdot r_k$
  - $r_{k+1} = b - Ax_{k+1}$
  - $k \leftarrow k + 1$

Custo:

- 2 produtos escalares
- 2 produto matriz-vetor
- 1 saxpy

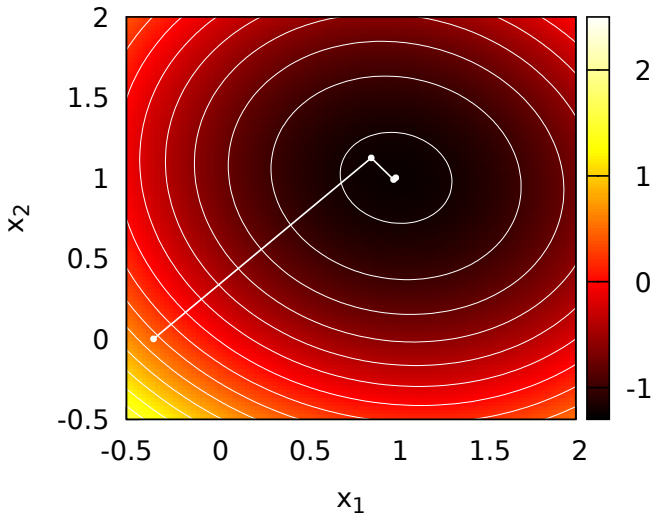
Total: $\mathcal{O}(n^2)$

Exemplo 1

$$\begin{bmatrix} 1.04 & 0.10 \\ 0.10 & 1.26 \end{bmatrix} x = \begin{bmatrix} 1.14 \\ 1.36 \end{bmatrix}$$

$$x_* = (1, 1)^T$$

$$\Lambda = \{1.0, 1.3\}$$

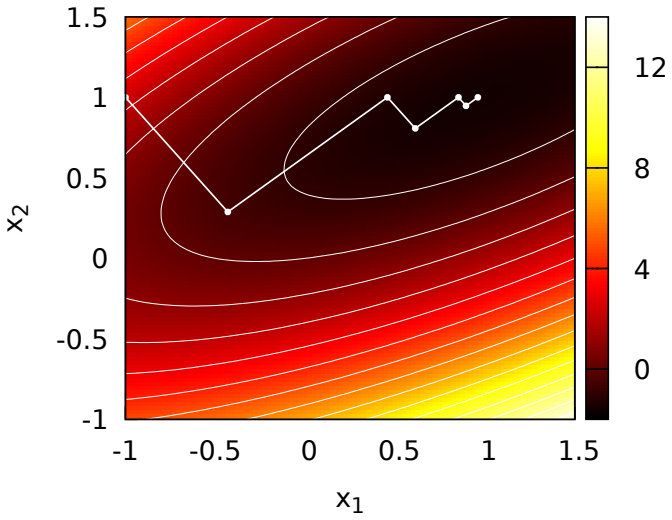


Exemplo 2

$$\begin{bmatrix} 2.00 & -2.50 \\ -2.50 & 6.25 \end{bmatrix} x = \begin{bmatrix} -0.50 \\ 3.75 \end{bmatrix}$$

$$x_* = (1, 1)^T$$

$$\Lambda = \{0.6, 3.6\}$$



Propriedades

- Computacionalmente barato
- Só depende de saber computar Ax
- Pode ser bem lento
- Velocidade depende da relação entre os autovalores de A

Teorema de Schur

UNICAMP

Teorema

Se $A \in \mathbb{C}^{n\times n}$, então existem $U \in \mathbb{C}^{n\times n}$, unitária, e $T \in \mathbb{C}^{n\times n}$, triangular superior, tais que

$$U^* A U = T.$$

Teorema de Schur: demonstração (1)

A prova é feita por indução em n.

(I) Para $n = 1$, $a = 1 \cdot a \cdot 1$.

(II) Suponha que o resultado vale para $n = k - 1$ (H.I.)

Seja (λ, v) um autopar de A, com $\|v\|_2 = 1$. Seja $U_1 = [v\ W] \in \mathbb{C}^{n \times n}$ uma matriz cujas colunas formam uma base ortonormal de $\mathbb{C}^n$.

Definindo $A_1 = U_1^* A U_1$, então

$$A_1 = \begin{bmatrix} v \\ W \end{bmatrix}^* A \begin{bmatrix} v & W \end{bmatrix} = \begin{bmatrix} v^*Av & v^*AW \\ W^*Av & W^*AW \end{bmatrix} = \begin{bmatrix} \lambda & v^*AW \\ 0 & W^*AW \end{bmatrix}$$

Teorema de Schur: demonstração (2)

$$A_1 = U_1^* A U_1 = \begin{bmatrix} \lambda & v^* AW \\ 0 & W^* AW \end{bmatrix}$$

Como $A_2 = W^* AW$ é de ordem $k-1$, pela H.I., existem $\hat{U}$ e $\hat{T}$ tais que

$$\hat{U}^* A_2 \hat{U} = \hat{T}.$$

Definindo $U_2 = \begin{bmatrix} 1 & 0 \\ 0 & \hat{U} \end{bmatrix}$, temos que

$$U^* A U = U_2^* (U_1^* A U_1) U_2 = \begin{bmatrix} 1 & 0 \\ 0 & \hat{U}^* \end{bmatrix} \begin{bmatrix} \lambda & v^* AW \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \hat{U} \end{bmatrix} = \begin{bmatrix} \lambda & \cdots \\ 0 & \hat{T} \end{bmatrix} = T$$

Matrizes simétricas: propriedade I

UNICAMP

Teorema

Se $A = A^T \in \mathbb{R}^{n \times n}$, então os autovalores A são reais.

Se v é autovetor de A associado ao autovalor λ, então

$$\bar{\lambda}\langle v, v\rangle = \langle Av, v\rangle = \langle v, Av\rangle = \lambda\langle v, v\rangle.$$

Logo $\lambda = \bar{\lambda}$, e portanto $\lambda \in \mathbb{R}$.

Lembre que:

- $\langle u, v\rangle = u^* v$
- $\langle \alpha u, v\rangle = \bar{\alpha} u^* v = \bar{\alpha}\langle u, v\rangle$
- $\langle u, \alpha v\rangle = \alpha u^* v = \alpha\langle u, v\rangle$

Matrizes simétricas: propriedade II

UNICAMP

Teorema

Se $A = A^T \in \mathbb{R}^{n \times n}$ e λ é autovalor de A, existe autovetor real associado.

Se $v \in \mathbb{C}^n$ é autovetor de A, então $\bar{v}$ também é, dado que

$$A\bar{v} = \overline{Av} = \overline{\lambda v} = \lambda \bar{v}.$$

Defina $u = (v + \bar{v})$. Logo $u \in \mathbb{R}^n$.

Se $u \neq 0$, então u é autovetor real associado a λ.

Se $u = 0$, então v é puramente imaginário, e portanto iv é autovetor real associado a λ.

Matrizes simétricas: propriedade III

UNICAMP

Teorema

Se $A = A^T \in \mathbb{R}^{n\times n}$ e λ_1 e λ_2 são autovalores distintos de A, então os autovetores associados são ortogonais.

Se v_1 e v_2 são os autovetores de A, então

$$\lambda_1\langle v_1, v_2\rangle = \langle \lambda_1 v_1, v_2\rangle = \langle Av_1, v_2\rangle = \langle v_1, Av_2\rangle = \langle v_1, \lambda_2 v_2\rangle = \lambda_2\langle v_1, v_2\rangle$$

Como $\lambda_1 \neq \lambda_2$, $\langle v_1, v_2\rangle = 0$.

Matrizes simétricas: propriedade IV

UNICAMP

Teorema

Se $A = A^T \in \mathbb{R}^{n \times n}$ é simples, ou seja, tem n autovalores distintos, então há uma base ortonormal para $\mathbb{R}^n$ formada por autovetores de A.

Se A é simples, há n autovetores, associados aos n autovalores. Pelo teorema anterior, são todos ortogonais entre si. Normalizando-os, temos uma base de autovetores ortonormais.

Decomposição espectral: demonstração

Teorema

Se $A = A^T \in \mathbb{R}^{n\times n}$, então existe $U \in \mathbb{R}^{n\times n}$, ortogonal, e $\Lambda \in \mathbb{R}^{n\times n}$, diagonal, tais que

$$A = U\Lambda U^T.$$

Pelo Teorema de Schur, existe U e T tais que

$$U^* A U = T$$

Como U é escolhida sempre tomando um autovetor de uma matriz simétrica, é sempre possíve escolher autovetores reais. Logo, temos

$$T = U^T A U = (U^T A U)^T = T^T.$$

Como $T = T^T$ e T é triangular superior, temos que T deve ser diagonal.

Relembrando o método de Jacobi

$$x_{k+1} = D^{-1}(D - A)x_k + D^{-1}b$$

Como não podemos medir $e_k = x_k - x^*$, usamos o residuo, $r_k = b - Ax_k$ como critério para convergência.

Veja que

$$r_k = b - Ax_k = Ax^* - Ax_k = -Ae_k.$$

Reescalando e reescrevendo

Considere que $D = I$ (se não for, basta reescalar A e b). Assim, $B_J = (I - A)$ e

$$r_k = b - Ax_k = b + x_k - Ax_k - x_k = b + B_J x_k - x_k = x_{k+1} - x_k,$$

ou seja

$$x_{k+1} = x_k + r_k.$$

Multiplicando por $-A$ e somando b a ambos os lados, temos

$$r_{k+1} = r_k - Ar_k = B_J r_k.$$

Resumo

Reescalando A por D^{-1}, temos

$$B_J \equiv I - A$$

$$e_k \equiv x_k - x^*$$

$$r_k \equiv b - Ax_k = -Ae_k$$

$$x_{k+1} \equiv B_J x_k + b = x_k + r_k$$

$$r_{k+1} = B_J r_k = (I - A) r_k$$

Espaço de Krylov

UNICAMP

Teorema

$$r_k = p_k(A)r_0 \in \text{span}\{r_0, Ar_0, \ldots, A^k r_0\},$$

onde $p_k(z) = (1-z)^k$ é um polinômio de grau exatamente k. Além disso,

$$x_k - x_0 \in \text{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\} \equiv \mathcal{K}_k.$$

Exercício: demonstre este resultado, por indução em k.

O espaço $\mathcal{K}_k$ é dito o k-ésimo espaço de Krylov associado a A e r_0. Repare que o espaço de Krylov de ordem k é gerado por k vetores. Entretanto, não sabemos (ainda) se esses k vetores são linearemente independentes e portanto não podemos (ainda) afirmar nada sobre a dimensão do $\mathcal{K}_k$.

Como melhorar Jacobi?

O k-ésimo iterando produzido pelo método de Jacobi pertence a $\mathcal{K}_k$,

mas será que este x_k é a melhor aproximação para a solução do sistema linear em $\mathcal{K}_k$?

Queremos um método tal que na iteração k, x_k minimize uma medida de erro sobre o espaço afim

$$x_0 + \mathcal{K}_k,$$

onde x_0 é aproximação inicial e o espaço de Krylov $\mathcal{K}_k$ é

$$\mathcal{K}_k = \text{span}\left\{r_0, Ar_0, \ldots, A^{k-1}r_0\right\}$$

e $r = b - Ax$.

Método de gradientes conjugados

- A, uma matriz simétrica e definida positiva
- $\|x\|_A = \sqrt{x^T Ax}$ (norma A)
- O iterando x_k minimiza

$$q(x) = \frac{1}{2}x^T Ax - x^T b,$$

sobre o espaço afim $x_0 + \mathcal{K}_k$.

Propriedade de minimização

$$q(x) = \frac{1}{2}x^T Ax - x^T b \quad \Rightarrow \quad \nabla q(x) = Ax - b$$

Se x^* minimiza $q(x)$ em $\mathbb{R}^n$, então x^* é solução de $Ax = b$.

Propriedade de minimização

UNICAMP

Lema

Seja $S \subset \mathbb{R}^n$. Se x_k minimiza q em S então x_k também minimiza $\|x^* - x\|_A = \|r\|_{A^{-1}}$.

$$\begin{aligned}\|x^* - x\|_A^2 &= (x^* - x)^T A(x^* - x) \\ &= x^T Ax - x^T Ax^* - (x^*)^T Ax + (x^*)^T Ax^* \\ &= x^T Ax - 2x^T b + (x^*)^T Ax^* \\ &= 2q(x) + (x^*)^T Ax^*\end{aligned}$$

$$\|x^* - x\|_A^2 = [A(x^* - x)]^T A^{-1} [A(x^* - x)] = [b - Ax]^T A^{-1}[b - Ax] = \|b - Ax\|_{A^{-1}}^2$$

Consequência

Se $S = x_0 + \mathcal{K}_k$, então

$$\|x^* - x_k\|_A \leq \|x^* - w\|_A, \text{ para todo } w \in x_0 + \mathcal{K}_k.$$

Assim,

$$x^* - w = x^* - \left(x_0 + \sum_{j=0}^{k-1} c_j A^j r_0 \right) = (x^* - x_0) - \sum_{j=0}^{k-1} c_j A^{j+1}(x^* - x_0) = p(A)(x^* - x_0),$$

onde

$$p(z) = 1 - \sum_{j=0}^{k-1} c_j z^{j+1}.$$

Na identidade acima, usamos que

$$A^j r_0 = A^j(b - Ax_0) = A^j(Ax^* - Ax_0) = A^{j+1}(x^* - x_0).$$

Consequência

Se x_k minimiza o erro em $x_0 + \mathcal{K}_k$ na norma $\|\cdot\|_A$, então

$$\|x^* - x_k\|_A = \min_{p \in \mathcal{P}_k,\, p(0)=1} \|p(A)(x^* - x_0)\|_A,$$

onde $\mathcal{P}_k$ é o conjunto dos polinômios de grau k.

Potências de uma matriz

Como A é uma matriz s.d.p., pelo teorema espectral,

$$A = U\Lambda U^T.$$

Logo

$$A^j = \left(U\Lambda U^T\right)\left(U\Lambda U^T\right)\cdots\left(U\Lambda U^T\right) = U\Lambda^j U^T.$$

Portanto,

$$p(A) = Up(\Lambda)U^T.$$

Definindo $A^{1/2} = U\Lambda^{1/2}U^T$, então

$$\|x\|_A^2 = x^T Ax = x^T U\Lambda^{1/2}\Lambda^{1/2}U^T x = x^T U\Lambda^{1/2}U^T U\Lambda^{1/2}U^T x = \|A^{1/2}x\|_2^2.$$

Propriedade de minimização

$$\|p(A)u\|_A = \|A^{1/2}p(A)u\|_2 \leq \|p(A)\|_2\|A^{1/2}u\|_2 = \|p(A)\|_2\|u\|_A.$$

Com isso

$$\|x^* - x_k\|_A \leq \|x^* - x_0\|_A \min_{p\in\mathcal{P}_k,\, p(0)=1} \|p(\Lambda)\|_2$$

Como $p(\Lambda)$ é diagonal com os autovalores de A, temos que

$$\|p(\Lambda)\|_2 = \max_{z\in\sigma(A)} |p(z)|.$$

Para ver que $\|p(A)\|_2 = \|p(\Lambda)\|_2$, precisamos conhecer a propriedade de que a norma euclidiana não é alterada por transformações ortogonais. Se U é uma matriz ortogonal, então

$$\|x\|_2^2 = x^T x = x^T U^T U x = \|Ux\|_2^2.$$

Sendo assim,

$$\|p(A)\|_2 = \|Up(\Lambda)U^T\|_2 = \|p(\Lambda)\|_2.$$

Além disso, precisamos também saber que a norma 2 de uma matriz diagonal é o maior valor em módulo dos elementos da diagonal da matriz. Fica como exercício mostrar esse fato.

Resultados

Teorema

Para A s.d.p. de ordem n, o método de Gradientes Conjugados encontra a solução do sistema $Ax = b$ em até n iterações.

Teorema

Se b é combinação linear de k autovetores de A, então o método de Gradientes Conjugados partindo de $x_0 = 0$ encontra a solução do sistema linear em até k iterações.

Teorema

Se A tem k autovalores distintos, então o método de Gradientes Conjugados encontra a solução do sistema linear em até k iterações.

Por conta desses resultados, podemos inclusive interpretar o método de Gradientes Conjugados como um método direto, visto que após uma quantidade determinadas de passos temos a solução exata do sistema.

Entretanto, para sistemas lineares de grande porte, aguardar n iterações pode ser proibitivo. Desta forma, é mais usual entender o método de Gradientes conjugados como um método iterativo e pensar então em critérios de parada que garantam uma aproximação aceitável da solução.

Mas como?

UNICAMP

x_k minimiza $\|x^* - x\|_A$ em $x_0 + \mathcal{K}_k$

Apesar dos vários resultados, ainda não sabemos como determinar x_k.

Direção de busca

Tudo depende de conseguirmos de forma barata uma direção de busca $d_{k+1} \neq 0$ tal que

$$x_{k+1} = x_k + \alpha_{k+1} d_{k+1}.$$

Se d_{k+1} for conhecido, o escalar α_{k+1} ótimo é determinado, como no método do gradiente, pela imposição de que

$$\frac{d\phi(x_k + \alpha_{k+1} d_{k+1})}{d\alpha} = 0 \quad \Rightarrow \quad \alpha_{k+1} = \frac{d_{k+1}^T r_k}{d_{k+1}^T A d_{k+1}}.$$

Resíduos ortogonais

UNICAMP

Teorema

Seja A s.d.p. e sejam $\{x_k\}$ os iterandos de Gradientes Conjugados. Então

$$r_k^T r_j = 0, \text{ para todo } 0 \leq j < k.$$

Se x_k minimiza ϕ em $x_0 + \mathcal{K}_k$, então para qualquer $y \in \mathcal{K}_k$

$$\frac{d\phi(x_k + ty)}{dt} = \nabla\phi(x_k + ty)^T y = 0$$

em $t = 0$. Como $\nabla\phi(x) = Ax - b = -r$, temos que

$$\nabla\phi(x_k)^T y = -r_k^T y = 0, \text{ para todo } y \in \mathcal{K}_k.$$

Como $r_j \in \mathcal{K}_k$, se $j < k$, fica demonstrado o resultado.

Corolário

Será que o método de Gradientes Conjugados por estagnar ($x_{k+1} = x_k$)?

Se $x_{k+1} = x_k$, então $r_{k+1} = r_k$. Com isso

$$\|r_k\|_2^2 = r_k^T r_k = r_{k+1}^T r_k = 0,$$

por força do teorema anterior. Se $r_k = 0$, então $x_k = x^*$.

Direções A-conjugadas

UNICAMP

Teorema

Sejam A s.d.p. e $\{x_k\}$ os iterados de Gradientes Conjugados. Se $x_k \neq x^*$, então $x_{k+1} = x_k + \alpha_{k+1} d_{k+1}$ e d_{k+1} é determinado a menos de uma constante por

$$d_{k+1} \in \mathcal{K}_{k+1} \quad \text{e} \quad d_{k+1}^T A y = 0, \text{ para todo } y \in \mathcal{K}_k.$$

Direções A-conjugadas: demonstração

Para todo $y \in \mathcal{K}_k$, como $\mathcal{K}_k \subset \mathcal{K}_{k+1}$,

$$\begin{aligned} 0 = \nabla\phi(x_{k+1})^T y &= [Ax_{k+1} - b]^T y \\ &= [A(x_k + \alpha_{k+1} d_{k+1}) - b]^T y \\ &= [Ax_k - b]^T y + \alpha_{k+1} d_{k+1}^T Ay \\ &= \nabla\phi(x_x)^T y + \alpha_{k+1} d_{k+1}^T Ay \\ &= \alpha_{k+1} d_{k+1}^T Ay \end{aligned}$$

Ou seja d_{k+1} é A-ortogonal a $\mathcal{K}_k$. Como $x_{k+1} \in \mathcal{K}_{k+1}$, temos que $d_{k+1} \in \mathcal{K}_{k+1}$.

Definido o produto interno

$$\langle x, y \rangle = x^T Ay,$$

temos que d_{k+1} é ortogonal (neste produto interno) ao espaço $calK_k$. Essa propriedade também é chamada de A-conjugação.

Direção d_{k+1}

Sabemos que

- $\mathcal{K}_k = \text{span}\{r_0, r_1, \ldots, r_{k-1}\}$
- $d_{k+1} \in \mathcal{K}_{k+1}$
- d_{k+1} é A-ortogonal a $\mathcal{K}_k$

Então

$$d_{k+1} = r_k + w_k, \text{ para algum } w_k \in \mathcal{K}_k.$$

Direção d_{k+1}

UNICAMP

Teorema

Seja A s.d.p. e suponha que $r_k \neq 0$. Definindo $d_0 = 0$, então

$$d_{k+1} = r_k + \beta_{k+1} d_k, \text{ para algum } \beta_{k+1}, \ k \geq 0.$$

Algoritmo de Gradientes Conjugados

Dados A, $n \times n$, s.d.p, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $\epsilon > 0$ e K

- $r \leftarrow b - Ax$, $\rho_0 = \|r\|_2^2$, $k \leftarrow 1$
- Enquanto $\sqrt{\rho_{k-1}} > \epsilon \|b\|_2$ e $k < K$, repita:
  - Se $k = 1$: $d \leftarrow r$

    caso contrário: $\beta \leftarrow \rho_{k-1}/\rho_{k-2}$ e $d \leftarrow r + \beta d$
  - $w \leftarrow Ad$
  - $\alpha \leftarrow \rho_{k-1}/(d^T w)$
  - $x \leftarrow x + \alpha d$
  - $r \leftarrow r - \alpha w$
  - $\rho_k \leftarrow \|r\|_2^2$
  - $k \leftarrow k + 1$

Custo computacional:

- 1 produto matriz-vetor
- 2 produtos internos
- 3 saxpy

Sistemas não lineares

Queremos agora resolver o sistema não linear

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ \vdots & \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{cases}$$

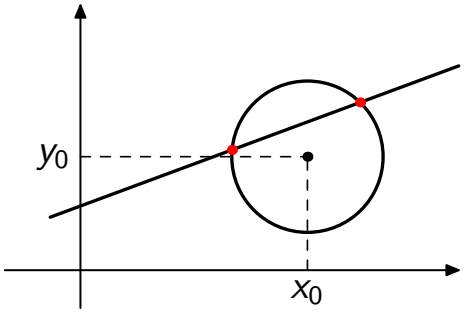
A extensão do problema de encontrar uma solução de uma única equação real é procurar uma solução para um sistema de equações não lineares. As variáveis agora são $x_1, x_2, \ldots, x_n$. Para não confundir (ou já confundindo), o subíndice j em x_j indica a j-ésima variável e não mais a aproximação computada na j-ésima iteração. Para discriminar iterações, utilizaremos índices acima, por exemplo $x_3^{(2)}$ representa a aproximação para a variável x_3 computada na segunda iteração.

Cada função f_j que compõe o sistema é uma função escalar, ou seja, $f_j : \mathbb{R}^n \to \mathbb{R}$, que a cada $(x_1, x_2, \ldots, x_n) \mapsto f(x_1, x_2, \ldots, x_n) \in \mathbb{R}$.

Atenção: Na discussão que se segue, vamos nos restringir a sistema com igual número de equações e variáveis.

Exemplo

$$\begin{cases} (x - x_0)^2 + (y - y_0)^2 - 1 &= 0 \\ ax + by + c &= 0 \end{cases}$$



Neste exemplo, as incógnitas são x e y, e x_0, y_0, a, b e c são parâmetros supostamente conhecidos. A primeira equação é não linear, representando um círculo, enquanto que a segunda equação é linear e descreve uma reta no plano. Basta que uma das equações seja não linear para que o sistema seja dito um sistema não linear.

Na configuração exemplificada na figura o sistema admite duas soluções, os pontos de intersecção da reta com o círculo. Caso a reta tangenciasse o círculo, haveria uma única solução. Ainda seria possível um terceiro caso em que a reta não cruzasse o círculo e neste caso o sistema seria sem solução.

Características

- Pode ter ou não solução
- Pode ter uma, algumas ou infinitas soluções
- Não é tão simples localizar soluções

Pelo exemplo anterior vimos que um sistema não linear pode ou não ter soluções. Quando o sistema tem solução, a quantidade de soluções pode variar de uma única a infinitas, passando inclusive por uma quantidade finita (maior que um). Isso é diferente do que acontece com um sistema linear, por exemplo, onde há apenas três situações possíveis: sem solução, solução única, ou infinitas soluções.

Também é muito difícil localizar regiões do domínio que seguramente contenham soluções. Como o sistema tem várias equações, não é possível avaliar o sinal de cada uma delas para decidir sobre a existência ou não de soluções.

Em notação vetorial,

$$\begin{aligned} x \in \mathbb{R}^n, \qquad & x = (x_1, x_2, \ldots, x_n)^T, \text{ e} \\ F : \mathbb{R}^n \to \mathbb{R}^n, \quad & F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T. \end{aligned}$$

Queremos encontrar $x^* \in \mathbb{R}^n$ tal que

$$F(x^*) = 0$$

Hipótese: F é diferenciável

A maneira de simplificar a notação e poder perceber as semelhanças e diferenças entre o problema unidimensional e multidimensional, é reescrever um sistema não linear usando notação vetorial.

1-D	n-D
$x \in \mathbb{R}$	$x \in \mathbb{R}^n$
$f : \mathbb{R} \to \mathbb{R}$	$F : \mathbb{R}^n \to \mathbb{R}^n$
Queremos x^* tal que:	
$f(x^*) = 0$	$F(x^*) = 0$

Exemplo

$$\begin{cases} x_1^2 - e^{-x_1 x_2} &= 0 \\ x_1 x_2 + \sin x_1 &= 0 \end{cases}$$

Neste caso,

$$F(x) = \begin{pmatrix} x_1^2 - e^{-x_1 x_2} \\ x_1 x_2 + \sin x_1 \end{pmatrix}$$

Neste caso, $f_1(x) = x_1^2 - e^{-x_1 x_2}$ e $f_2(x) = x_1 x_2 + \sin x_1$ são as componentes de $F(x)$.

Taylor $\Rightarrow$ Newton

$$F(x+s) = F(x) + J(x)s + \mathcal{O}(\|s\|^2)$$

$J(x)$ é a matriz Jacobiana de F

$$J(x) = \begin{pmatrix} \nabla^T f_1(x) \\ \nabla^T f_2(x) \\ \vdots \\ \nabla^T f_n(x) \end{pmatrix}$$

Em 1-D:

$$f(x+s) = f(x) + f'(x)s + \mathcal{O}(s^2), \qquad f'(x)s \approx f(x+s) - f(x)$$

Em n-D:

$$F(x+s) = F(x) + J(x)s + \mathcal{O}(\|s\|^2), \qquad Js \approx F(x+s) - F(x)$$

Exemplo

$$F(x) = \begin{pmatrix} x_1^2 - e^{-x_1 x_2} \\ x_1 x_2 + \sin x_1 \end{pmatrix}$$

$$J(x) = \begin{pmatrix} 2x_1 + x_2 e^{-x_1 x_2} & x_1 e^{-x_1 x_2} \\ x_2 + \cos x_1 & x_1 \end{pmatrix}$$

Observe as linhas da matriz Jacobiana são os gradientes das componentes de F.

Método de Newton para sistemas

$$F(x+s) = F(x) + J(x)s + \mathcal{O}(\|s\|^2)$$

$$F(x+s) \approx F(x) + J(x)s$$

Impondo que $F(x) + J(x)s = 0$, temos que

$$J(x)s = -F(x)$$

Nova aproximação

$$x + s = x - J(x)^{-1}F(x)$$

O método de Newton para sistemas é construído de maneira análoga ao método de Newton para uma equação escalar, ou seja, considerando-se a aproximação de Taylor de primeira ordem apenas.

Em cada iteração, um sistema linear deve ser resolvido para descobrir s, denominado passo. Esse passo é então somado à aproximação atual para obter a nova aproximação.

Algoritmo

- Seja $x^{(0)}$ uma aproximação razoável de x^*
- Para $k = 0, 1, 2, \ldots$
  - Se $J(x^{(k)})$ for não-singular, resolver $J(x^{(k)})s^{(k)} = -F(x^{(k)})$
  - $x^{(k+1)} \leftarrow x^{(k)} + s^{(k)}$

Repare a semelhança entre esse algoritmo e o algoritmo para o método de Newton no caso de uma única equação escalar.

Exemplo: ponto inicial

$$F(x) = \begin{pmatrix} x_1^2 - e^{-x_1 x_2} \\ x_1 x_2 + \sin x_1 \end{pmatrix} \qquad J(x) = \begin{pmatrix} 2x_1 + x_2 e^{-x_1 x_2} & x_1 e^{-x_1 x_2} \\ x_2 + \cos x_1 & x_1 \end{pmatrix}$$

Se $x^{(0)} = (2, 1)^T$, então

$$F(x^{(0)}) = \begin{pmatrix} 3.8647 \\ 2.9093 \end{pmatrix} \qquad J(x^{(0)}) = \begin{pmatrix} 4.1353 & 0.2707 \\ 0.5839 & 2.0000 \end{pmatrix}$$

$$\|F(x^{(0)})\|_\infty = 3.8647$$

Arbitrariamente foi escolhido como ponto inicial $x^{(0)} = (2, 1)^T$. Nesse ponto, avaliar a função F e da matriz Jacobiana J. Observe que a $\|F(x^{(0)})\|$ da uma medida de quão perto $x^{(0)}$ está de satisfazer $F(x) = 0$.

Aqui usamos a norma infinito apenas por ser mais simples. Outras normas poderiam ser usadas também.

Exemplo: primeira iteração

$$\begin{pmatrix} 4.1353 & 0.2707 \\ 0.5839 & 2.0000 \end{pmatrix} s^{(0)} = - \begin{pmatrix} 3.8647 \\ 2.9093 \end{pmatrix}$$

$$s^{(0)} = (-0.8557, -1.2049)^T, \qquad x^{(1)} = x^{(0)} + s^{(0)} = (1.1443, -0.2049)^T$$

$$F(x^{(1)}) = \begin{pmatrix} 0.0453 \\ -0.6760 \end{pmatrix} \qquad \|F(x^{(1)})\|_\infty = 0.6760$$

Para progredir de $x^{(0)}$ a $x^{(1)}$ é necessário resolver o sistema linear que determina o passo $s^{(0)}$. Feito isso, obtemos a nova aproximação.

Repare que nesse novo ponto, o valor de $\|F(x^{(1)})\| < \|F(x^{(0)})\|$, indicando que houve melhora em satisfazer o sistema não linear.

Exemplo: segunda iteração

$$\begin{pmatrix} 2.0297 & 1.4466 \\ 0.2088 & 1.1443 \end{pmatrix} s^{(1)} = - \begin{pmatrix} 0.0453 \\ -0.6760 \end{pmatrix}$$

$$s^{(1)} = (0.4584, -0.6744)^T, \qquad x^{(2)} = x^{(1)} + s^{(1)} = (1.6026, -0.8793)^T$$

$$F(x^{(2)}) = \begin{pmatrix} -1.5239 \\ -0.4097 \end{pmatrix} \qquad \|F(x^{(2)})\|_\infty = 1.5239$$

Estranhamente, nesta segunda iteração parece que tivemos uma piora na aproximação visto que $\|F(x^{(2)})\| > \|F(x^{(1)})\|$.

Exemplo: terceira iteração

$$\begin{pmatrix} -0.3930 & 6.5589 \\ -0.9111 & 1.6027 \end{pmatrix} s^{(2)} = - \begin{pmatrix} -1.5239 \\ -0.4097 \end{pmatrix}$$

$$s^{(2)} = (0.0457, 0.2296)^T, \qquad x^{(3)} = x^{(2)} + s^{(2)} = (1.5569, -0.6497)^T$$

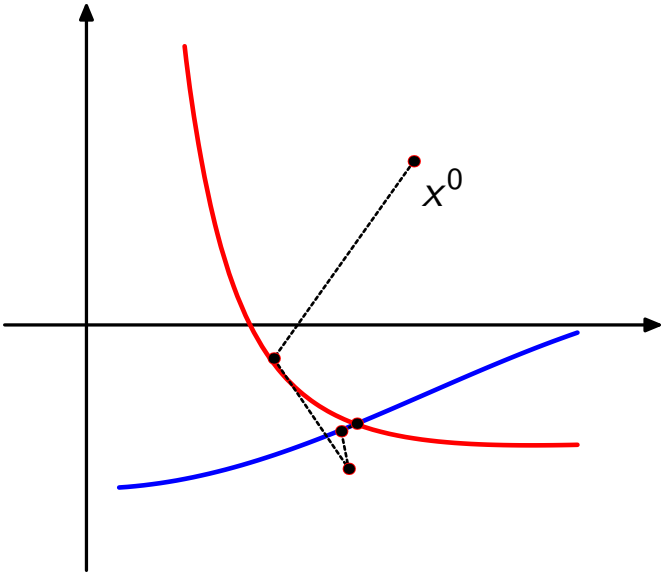
$$F(x^{(3)}) = \begin{pmatrix} -0.3256 \\ -0.0115 \end{pmatrix} \qquad \|F(x^{(3)})\|_\infty = 0.3256$$

Nesse exemplo, temos que

k	$\|F(x^{(k)})\|$	
0	3.8647	
1	0.6760	
2	1.5239	$\leftarrow$ aumentou!
3	0.3256	

Esse comportamento significa o quê? Dá para confiar que a sequência está convergindo? Pense em um exemplo em uma dimensão, isto é, um exemplo com uma única equação não linear, onde esse mesmo comportamento é observado.

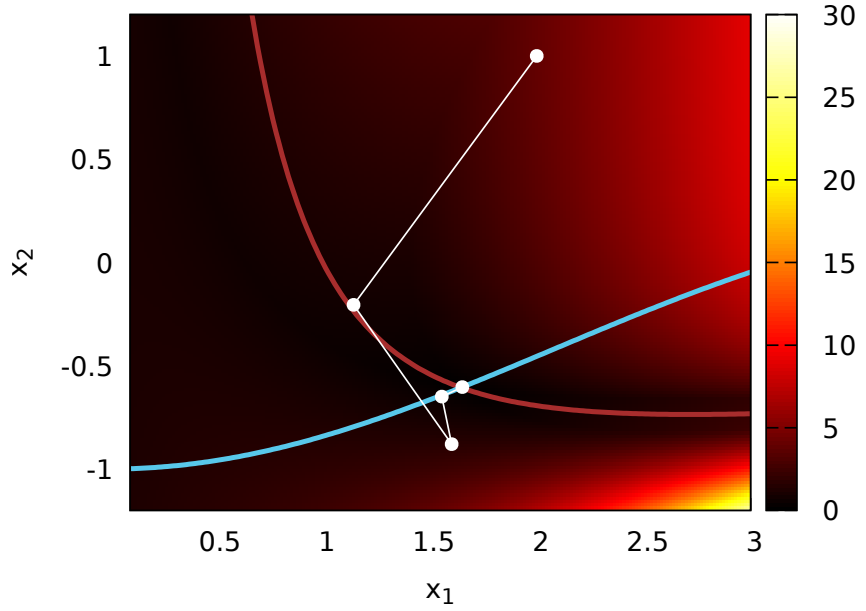
Exemplo: trajetória



Observe que neste exemplo, mesmo com o valor de função tendo aumentado, em todos os iterandos se aproximaram gradativamente da solução do sistema. De fato, a partir da terceira iteração é possível perceber que os iterandos devem estar dentro da região de convergência quadrática do método de Newton.

k	$\|F(x^{(k)})\|_\infty$	$\|x^{(k)} - x^*\|_\infty$	
0	3.8647	1.6057	
1	0.6760	0.5021	
2	1.5239	0.2734	
3	0.3256	0.0894	$\leftarrow$ conv. quadrática
4	0.0210	0.0061	
5	0.0001	0.0000	

Exemplo: trajetória



Nesse gráfico, as cores indicam o valor de $\|F(x)\|_\infty$. Agora fica mais claro ver que de $x^{(1)}$ para $x^{(2)}$ realmente houve um aumento no valor da norma infinito de $F(x)$, apesar de $x^{(2)}$ estar mais próximo de x^*.

Critérios de parada

- $\|s^{(k)}\| \leq \epsilon \|s^{(0)}\|$
- $\|F(x^{(k)})\| \leq \epsilon \|F(x^{(0)})\|$
- $\|F(x^{(k)})\| \leq \epsilon_1 \|F(x^{(0)})\| + \epsilon_2$

Os critérios de parada são totalmente análogos aos critérios utilizados no caso de uma única equação não linear.

Algoritmo

- Seja $x^{(0)}$ uma aproximação razoável de x^*
- Enquanto $\|F(x^{(k)})\| > \epsilon\|F(x^{(0)})\|$ e $k < K$
  - Se $J(x^{(k)})$ for não-singular, resolver $J(x^{(k)})s^{(k)} = -F(x^{(k)})$
  - $x^{(k+1)} \leftarrow x^{(k)} + s^{(k)}$
  - $k \leftarrow k + 1$

Subproblema – Sistema Linear

$$J(x^{(k)})s^{(k)} = -F(x^{(k)})$$

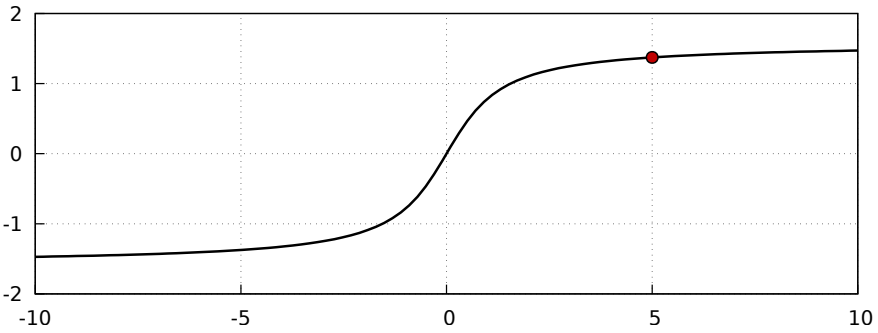
- Métodos diretos
- Métodos iterativos
  - Solução exata
  - Solução aproximada (Newton Inexato)

No caso do método de Newton para sistemas não lineares surge a questão de como resolver o sistema linear gerado em cada iteração. Esse problema é denominado de subproblema.

Para o subproblema, tanto métodos diretos quanto iterativos podem ser utilizados, e no caso de métodos iterativos, pode-se empregá-los até atingir convergência na solução ou para em uma solução aproximada. Neste caso deve-se apenas tomar o cuidado de exigir que a qualidade dessas soluções aproximadas aumentem a medida que progredimos nas iterações do método de Newton.

Falha em convergência

$$f(x) = \arctan(x)$$



$$5,\ -30.7,\ 1.4 \cdot 10^3,\ -3.2 \cdot 10^6,\ 1.6 \cdot 10^{13}, \ldots$$

Apesar da direção de Newton estar correta, apontandao para $x^* = 0$, o passo tomado ultrapassa em muito a posição do zero.

Critério de Armijo

Como a direção de Newton é $d \equiv -f(x)/f'(x)$, um passo nessa direção produz

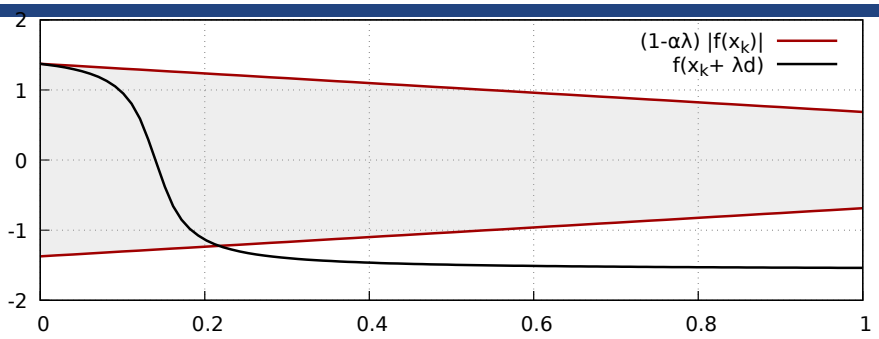
$$f(x + \lambda d) \approx f(x) + \lambda f'(x)d = f(x) - \lambda f(x) = (1 - \lambda)f(x)$$

O critério de Armijo exige que um decréscimo suficiente, ou seja, um λ tal que

$$|f(x + \lambda d)| < (1 - \alpha\lambda)|f(x)|,$$

para $\alpha \in (0, 1)$, fixo.

Backtracking



$$x_k = 5, \quad d = -f(x_k)/f'(x_k) = -35.7, \quad \alpha = 0.5$$

λ	1.00	$1/2$	$1/4$	$1/8$
$(1-\alpha\lambda)\lvert f(x_k)\rvert$	0.69	1.03	1.20	1.28
$\lvert f(x_k+\lambda d)\rvert$	1.53	1.49	1.32	0.49

Algoritmo com busca linear

- Seja x uma aproximação de x^*, $\epsilon \leftarrow \epsilon_r \|F(x^{(0)})\| + \epsilon_a$, $k \leftarrow 1$
- Enquanto $\|F(x)\| > \epsilon$ e $k < K$
  - Se $J(x)$ for não-singular, resolva $J(x)d = -F(x)$
  - $\lambda \leftarrow 1$
  - Enquanto $\|F(x + \lambda d)\| > (1 - \alpha\lambda)\|F(x)\|$, onde $\alpha \in (0, 1)$
    - $\lambda \leftarrow \sigma\lambda$, onde $\sigma \in (1/10, 1/2)$
  - $x \leftarrow x + \lambda d$
  - $k \leftarrow k + 1$

Para esse algoritmo, os parâmetros são as tolerâncias relativa e absoluta, ϵ_r e ϵ_a, o número máximo de iterações, K, e os parâmetro α e σ que controlam o critério de decréscimo suficiente de Armijo e como deve ser reduzido o passo de Newton durante o *backtracking*.

Como estamos supondo que o sistema linear $Jd = -F$ está sendo resolvido exatamente, é garantido que haja λ que satisfaça o critério de Armijo. Caso o sistema linear seja resolvido apenas de forma aproximada, isto pode não ser verdade. Para garantir que a direção de d seja de descida, deve ser imposta uma condição sobre sobre essa direção. Não vamos entrar nesse detalhe aqui.

UNICAMP

Teorema

Seja $x^{(0)} \in \mathbb{R}^n$ e $\alpha \in (0,1)$. Suponha que $x^{(k)}$ foram computados pelo algoritmo anterior, F é Lipschitz contínua, e $\{x^{(k)}\}$ e $\{\|J(x^{(k)})^{-1}\|\}$ são limitadas. Então $\{x^{(k)}\}$ converge para x^*, zero de F, além disso, a partir de algum passo suficientemente grande o passo cheio de Newton é sempre tomado ($\lambda = 1$) e a convergência é prescrita pelo resultado de convergência local.

Jacobiana

O passo caro e mais sensível é a resolução do sistema linear $J(x)d = -F(x)$.

O teorema anterior pode ser alterado para contemplar a solução aproximada do sistema linear, mas qualidade dessa solução pode comprometer a busca linear.

Falha na busca linear (passos pequenos demais) é um indício de que a aproximação da Jacobiana ou a resolução do sistema linear não estão boas.

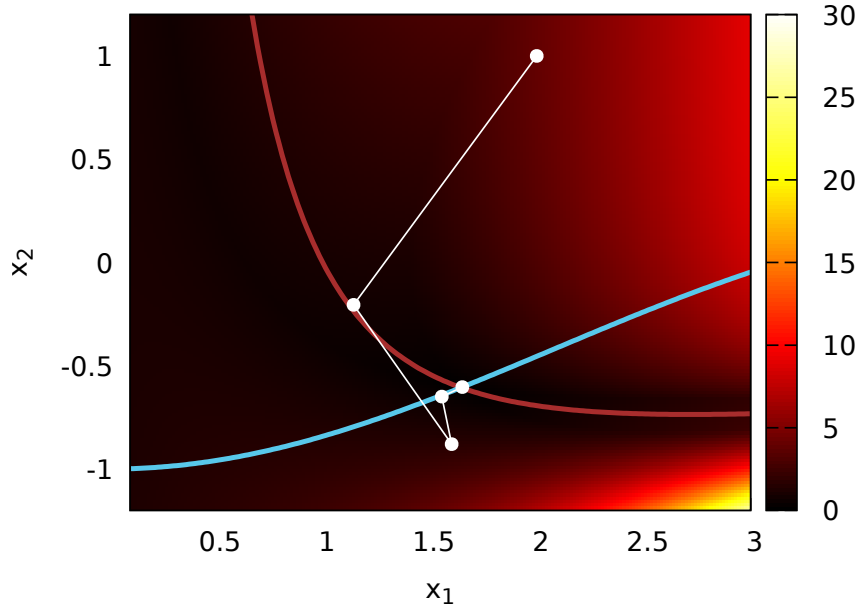
Jacobiana por diferenças finitas

Caso a Jacobina não seja conhecida, podemos aproximá-la por diferenças finitas. A j-ésima coluna de $J(x)$ pode ser aproximada por

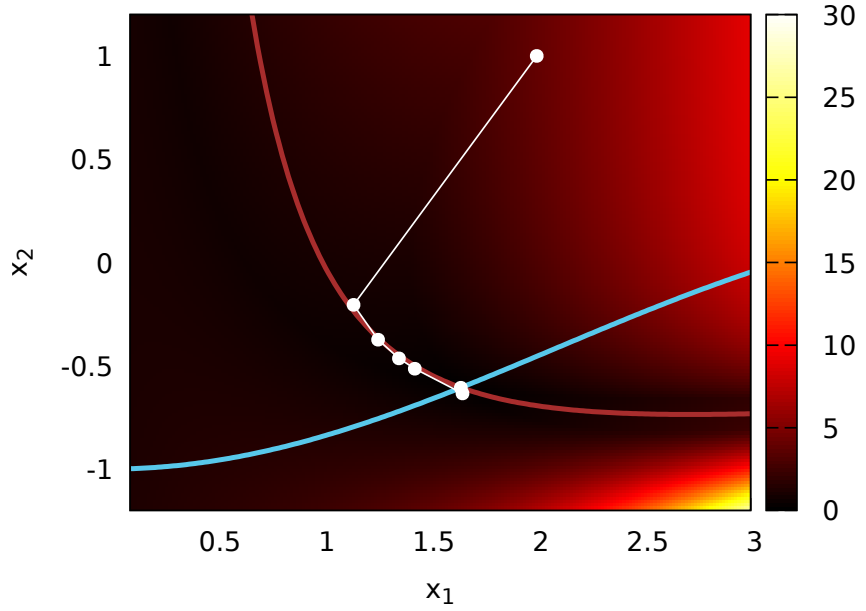
$$[J(x)]_j = \begin{cases} \dfrac{F(x + h\sigma_j e^j) - F(x)}{\sigma_j h}, & x_j \neq 0 \\ \dfrac{F(he_j) - F(x)}{h}, & x_j = 0 \end{cases}$$

onde $\sigma_j = \text{sign}(x_j) \cdot \max\{|x_j|, 1\}$ e h é um passo escolhido (tipicamente $h = 10^{-7}$) e e^j é o j-ésimo vetor da base canônica.

Exemplo: trajetória sem busca linear



Exemplo: trajetória com busca linear ($\alpha = 0.5$, $\sigma = 0.25$)



Neste exemplo (a mesma função já usada anteriormente), escolhemos como parâmetros para o algoritmo $\alpha = 0.5$, $\sigma = 0.25$, $\epsilon_a = \epsilon_r = 10^{-4}$. O método de Newton convergiu em 7 iterações. Observe abaixo como ficou a sequência de valores para a norma infinito em cada iteração

$$3.8647, 0.6760, 0.4816, 0.3479, 0.2545, 0.1192, 4.114 \cdot 10^{-3}, 2.342 \cdot 10^{-6}.$$

Perceba que a convergência quadrátrica se evidenciado a partir da quinta iteração.